MACHINE LEARNING NANODEGREE PROGRAMME

Project Report

David K. Njuguna
(dakn2005@gmail.com)
August 24th, 2018

I. Definition

Project Overview

Cash transfer programmes are designed to help alleviate the individuals from poverty[6]. These programmes are funded mostly by governments[5] under Social Protection Services[4], but are not limited to being government-funded.

Non-governmental organizations   can also operate some cash transfer programmes, depending with the social protection laws implemented in individual countries.

As an information systems practitioner, my major motivation of implementing this project is providing a model enabling NGOs with a quick way of determining eligibility of a household using selected features. This is because the process of determining eligibility can be very time consuming and costly, the after effect of this being communities in need having an unnecessarily lengthy wait time before aide arrives.

There are several machine learning algorithms techniques implemented for cash transfer programmes, but these focus solely on fraud detection, making the area of classifying households quite unique. The dataset contains a generalized set of features that many CT programmes dealing with households would use.

Problem Statement

Develop a model that can quickly, and accurately classify a household into two groups, given the features provided. These groups determine the household payment cycles, depending on the severity of the household.

Ideally the model should perform accurate household placements, while taking the least amount of time. For this purpose we will use supervised learning algorithm for this classification task. The three main algorithms are decisiontreeclassifier, SVM and XGBoost algorithm.

SVM is a generalized learning algorithm which can be used for both regression and classification problems, and is a good general algorithm for most problems. The main disadvantage is that it is computationally intensive, therefore will use more time compared to other algorithms. It also has parameters which are complex to tune i.e. the kernel trick involves setting a kernel depending on the type of data

Decisiontreeclassifier is a learning algorithm specific to classification problems. The main advantage of this algorithm is it requires relatively little effort from users for data preparation, and takes the least times for training compared to other algorithms. A major disadvantage for this method is overfitting if the parameters are not properly set

XGBoost stands for eXtreme Gradient Boosting, a relatively new learning algorithm which is an implementation of gradient boosted tree algorithms. It is generally a fast method compared to other methods of gradient boosting and bagged decision trees. Gradient boosting algorithms utilize the gradient descent algorithms to minimize loss when modelling

Metrics

This being a classification problem, we'll be using precision, recall and accuracy scores to check how well the algorithms perform at the task. The metrics are described below

tp = true positive, *tn* = true negative, *fn* = false negative, *fp* = false positive, *N* = dataset size, *B* = beta

Recall = $\frac{tp}{tp + fn}$ Precision = $\frac{tp}{tp + fp}$ Accuracy = $\frac{tp + tn}{N}$

$$F_B = (1 + B\hat{}2).\frac{precision.recall}{(B\hat{}2.precision) + recall}$$

Accuracy would be sufficient for a balanced dataset but looking at the data in the *exploration section*, we observe that the dataset is imbalanced. This being the case we'll also observe the resulting F-beta score.

For *B* < 1, we get a score that skews towards precision

For *B* > 1, we get a score that skews towards sensitivity (recall)

F-*beta* score is the harmonic mean of precision and recall.

II. Analysis

Data Exploration

Data exploration is performed under “Exploring the data” subject in the notebook. We check the type of variables represented which were categorized as categorical, continuous or discrete variables.

Data source: https://github.com/dakn2005/MLNDProject_CT/blob/master/mlnd-ds-2.csv.

To acquire/request for new data, you have to send a request form to the programme using http://hsnp.or.ke/images/mis/hsnp_data_request_form.pdf

We then check for total number of households present in the dataset; in the process checking for proportion of *group 1* vs *group 2* households. Initial observations are as follows:

Total number of records: 100,000
Households in group 1: 27,245
Households in group 2: 72,755
Percentage HH in group 1: 27.25%
Percentage HH in group 2: 72.76%

As observed from the target variable **HHGroup** percentages for group 1 and 2 above, the dataset is imbalanced.

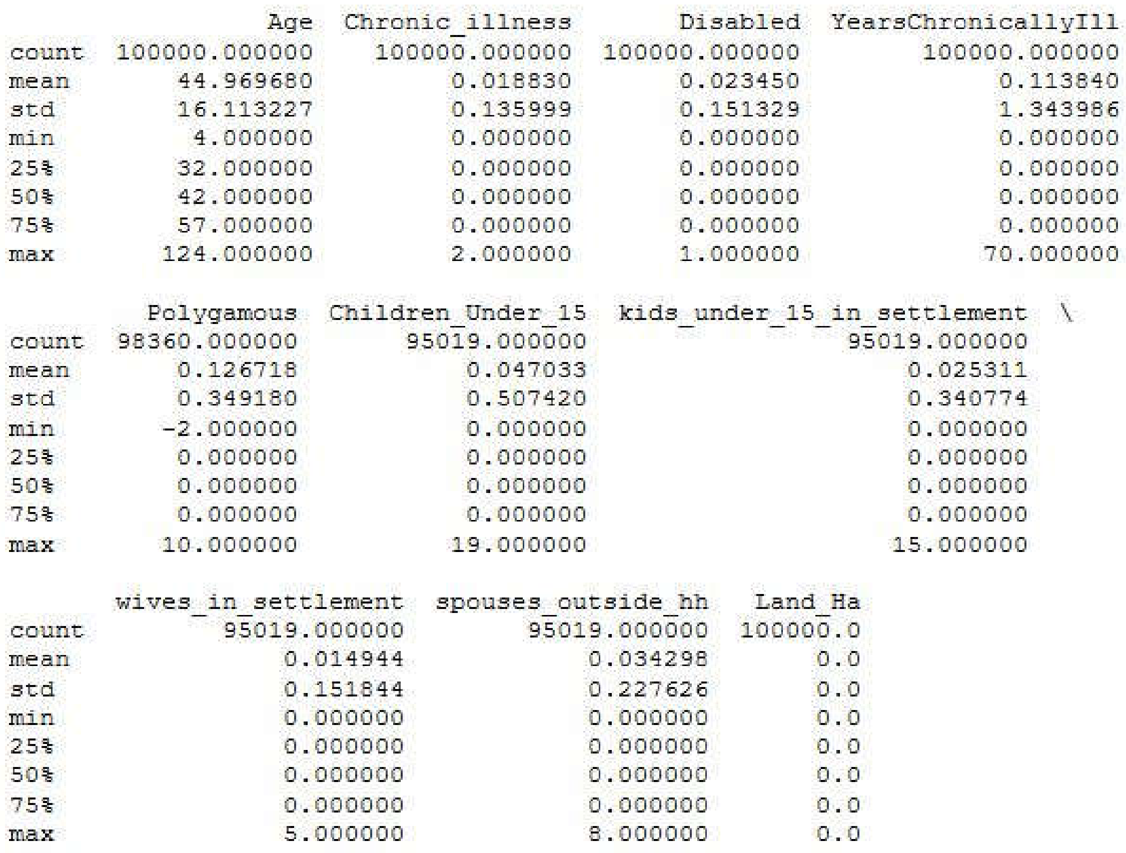
The dataset has the following features:

- **Gender**: Male, Female
- **Age**: continuous.
- **Attended_school**: Currently attending, Never, Yes but not now, SKIP
- **Work_last_7days** (had work at time of registration ?): SKIP, Worked for pay, On leave, Sick leave, Worked on own/family, business, Herding-Worked on own/family agri. holding, Seeking work, Doing nothing, Retired, Homemaker, Full-time, student, In capacitated, Other
- **Chronic_illness**: 0 - false, 1- true, 2-skip
- **Disabled**: 0- false, 1- true
- **YearsChronicallyIll**: continuous
- **WealthGroup**: Better Off, Middle, Poor, Very Poor
- **Resident_Provider**: SKIP, All live here all of the time, Some live here while others migrate, All migrate together-fully mobile, Inside this household, Outside household but inside compound-settlement, Outside this compound-settlement
- **Polygamous**: discrete
- **Children_Under_15**: discrete
- **kids_under_15_in_settlement**: discrete
- **wives_in_settlement**: discrete
- **spouses_outside_hh**: discrete
- **HHGroup**: categorical
- **Toilet**: Flush toilet, VIP latrine, Uncovered pit latrine, Covered pit latrine, Bucket/pan, Bush-None, Other

- **Drinking_water**: Piped water inside dwelling, Piped water into plot/yard, Public tap, Tube well/borehole with pump, Protected dug well, Protected spring, Rainwater collection, Unprotected dug well/springs, River, Lake, ponds or similar, Water truck/vendor, Bottled water, Other
- **Land_Ha**: discrete

Exploratory Visualization

We perform a description of the data to check on summary statistics i.e. the mean, standard deviation, percentiles, min and max values

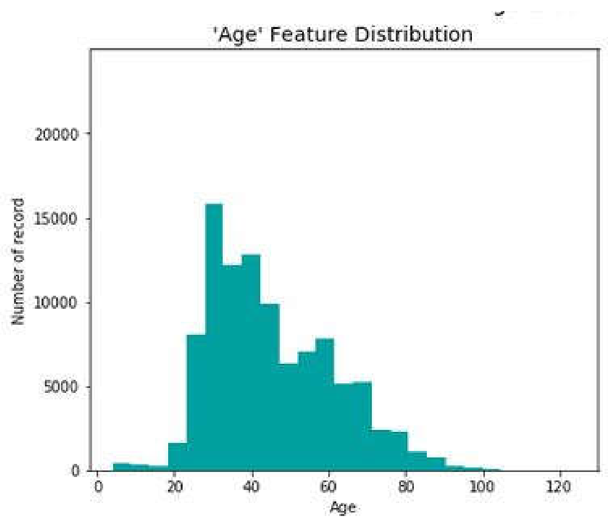


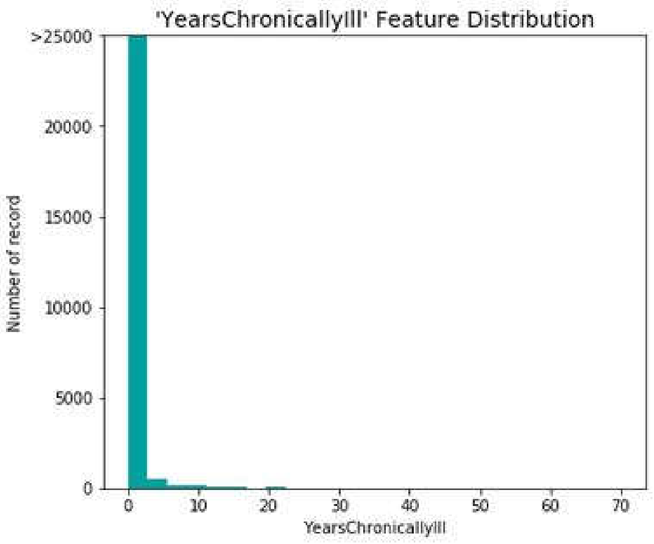
```
                 Age  Chronic_illness       Disabled  YearsChronicallyIll
count  100000.000000    100000.000000  100000.000000        100000.000000
mean       44.969680         0.018830       0.023450             0.113840
std        16.113227         0.135999       0.151329             1.343986
min         4.000000         0.000000       0.000000             0.000000
25%        32.000000         0.000000       0.000000             0.000000
50%        42.000000         0.000000       0.000000             0.000000
75%        57.000000         0.000000       0.000000             0.000000
max       124.000000         2.000000       1.000000            70.000000

        Polygamous  Children_Under_15  kids_under_15_in_settlement  \
count  98360.000000       95019.000000                 95019.000000
mean       0.126718           0.047033                     0.025311
std        0.349180           0.507420                     0.340774
min       -2.000000           0.000000                     0.000000
25%        0.000000           0.000000                     0.000000
50%        0.000000           0.000000                     0.000000
75%        0.000000           0.000000                     0.000000
max       10.000000          19.000000                    15.000000

       wives_in_settlement  spouses_outside_hh   Land_Ha
count         95019.000000        95019.000000  100000.0
mean              0.014944            0.034298       0.0
std               0.151844            0.227626       0.0
min               0.000000            0.000000       0.0
25%               0.000000            0.000000       0.0
50%               0.000000            0.000000       0.0
75%               0.000000            0.000000       0.0
max               5.000000            8.000000       0.0
```

I selected continuous features for visualization, which included *Age* and *YearsChronicallyIll*, checking for skewness to determine whether we will need perform transformation on these features.





YearsChronicallyIll feature is not statistically significant since the number of records having *YearsChronicallyIll > 0* is 1.54% of the total dataset, therefore did not perform log-transformation on this feature.

Checking the graph above and dataset description, the *Age* feature has a marginal standard deviation which would be resolved by performing normalization on numerical values, therefore did not log-transform this feature. The feature also follows a normal distribution as observed above.

Algorithms and Techniques

We tested three supervised algorithms, namely SVM, Decisiontreeclassifier and XGBoost classifier. We employ supervised learning techniques because we have a *target variable* to be predicted.

SVM - A generalized algorithm that can perform both regression and classification tasks, depending on the selected kernel. We set three major parameters/properties for this algorithm, namely kernel, degree or gamma (depending on the type of kernel selected) and a classification error constant C. C is a regularization parameter that controls the trade off between achieving a low training error and a low testing error i.e. the ability to generalize the classifier to unseen data.

$C \propto \frac{1}{M}$ M=margin, where large C reduces classification error, therefore classifies well, but might not generalize well (depending on unseen data). A low C increases the largest minimum margin, generalizing better at the cost of having higher classification errors

This algorithm works by reducing the lassification and margin errors. This is performed by c maximizing the margin where

$M \propto \frac{1}{E^2}$ where M = Margin, E = Error

therefore the larger the *largest minimum margin*, the lower the error. A kernel should be appropriately chosen depending on the type of data being presented i.e. data with many features (higher dimensions) should be presented using radial basis functions, which suits a bigger function space. If expecting a linear model, a polynomial or linear kernel would be used

DecisionTreeClassifier - A decisiontree algorithm specific to classification tasks. The three major parameters set are *max_depth, max_leaf_nodes, min_samples_split*.

This algorithm works by dividing the training data into groups, splitting according to each feature in the data. The split with the lowest cost is chosen, where a cost function is used which minimizes standard deviation. This process is recursive in nature, as groups formed are further subdivided.

The *max_depth* (maximum depth) and *max_leaf_nodes* parameters are set to determine when a decision tree should resolve (stop). These parameters are tuned using optimization methods like *gridsearch*

This algorithm is also known as the *greedy algorithm*, as it highly concentrates on lowering cost

XGBoost - Extreme Gradient Boosting. This algorithm uses decisiontree ensembles. These consist of a set of classification and regression trees, and work much like random forest algorithm, the major difference being in how they are trained. The major parameters to set are *objective, booster, learning_rate, n_estimators.*

Objective - specifies the a learning algorithm to be used according to the learning task.

Booster - the ensemble method to be used i.e. gradient boosted trees (gbtree)

Learning_rate - used to determine learning steps when performing gradient descent

n_estimators - number rounds to perform boosting

Benchmark

Developed a benchmark using *naive predictor* as the base model, with the assumptions that there no negative values (neither true negatives nor false negatives). Will check out the accuracy, f-score, recall and precision values. The resultant final model should give better scores especially on accuracy in comparison with this model.

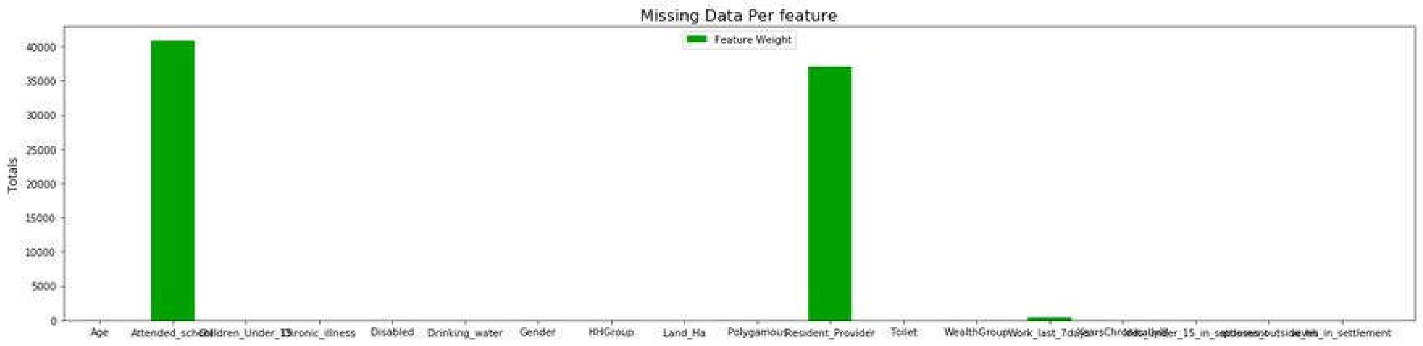
III. Methodology

Data Preprocessing

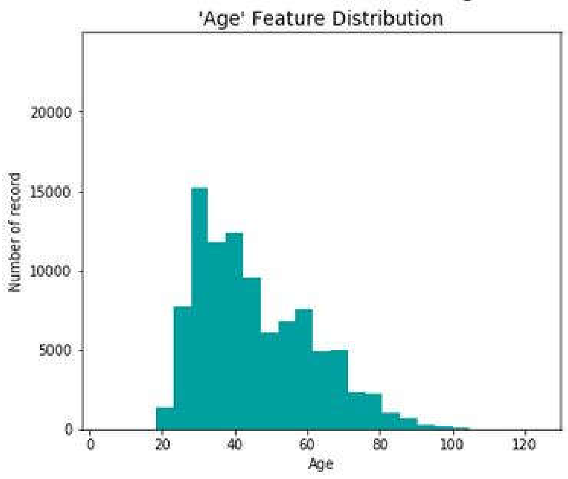
Several steps were taken during this stage:

First , I Investigated the features checking for missing data and presence of skewness in the dataset. Performing a description of the dataset using the describe function *data.describe(),* checking specifically at standard deviation for skewness on the specific feature. A high standard deviation translates to high skewness of data

Secondly, checking for missing data, I classified all records with a 'SKIP' entry as missing data. This was graphed as below. The most prominent features which were *Attended_school* and *Resident_provider* were dropped from the dataset



After expunging missing data, age distribution for below 20 years (age < 20 years) was no longer represented. This means data for minors (assuming age < 20 is a minor) had not been properly captured/reported



Thirdly, used the *MinMaxScaler* to normalize numerical features. Through normalization, skewness in the data by transforming to a normal distribution. Normalization is carried out on continuous variables.

Finally, Due to the fact that we cannot feed non-numeric values to the supervised learning algorithms we tested, we converted categorical data from non-numeric to numeric values using *one-hot encoding scheme* to accomplish this.

We also converted the target variable (HHgroup) from boolean values to numerical representation via a simple *replace* function. Note that **True** represents **group 1** and **False** represents **group 2**

Implementation

After performing the preprocessing steps above, the data is now ready for analysis. We start off by splitting the data into training and testing set.

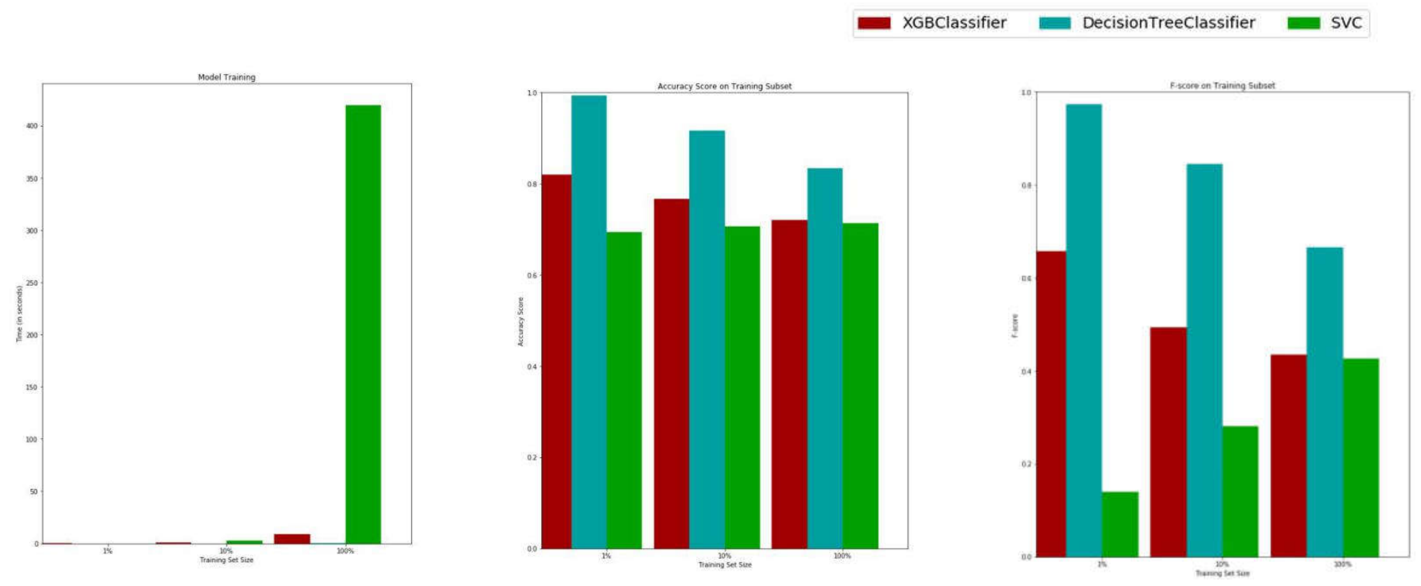
Next, we created a *naive predictor* as the benchmark model. As the name indicates, these indicator operates on some assumption which we'll have to make during initialization. In our case we used the assumption that there are no predicted negatives i.e. no True or false negatives. This gave us the following scores:

- ✧ Accuracy score: 0.2744
- ✧ F-score: 0.5514
- ✧ Recall: 1.0000
- ✧ Precision: 0.2744

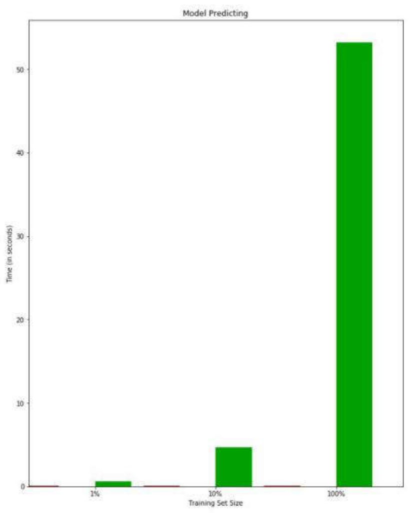
Using the three selected supervised algorithms, namely; decisiontreeclassifier, SVM and XGBoost, we proceeded to model evaluation on each of this. Graphing the results to check on performance per algorithm compared with the time taken by each algorithm, we obtained the graphs below:

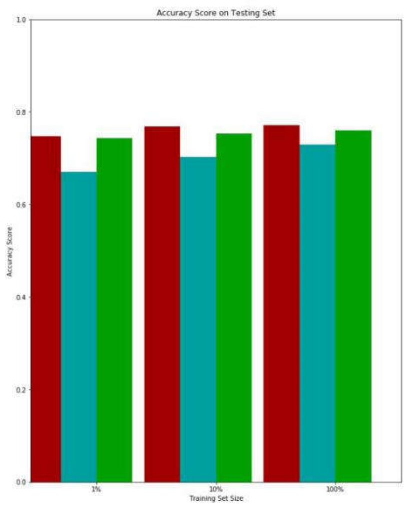
Using sample sizes ranging from 1% of the final dataset, to 10%, to full dataset (100%), to check the performance of these scores across different sizes of data. This checks on robustness of the algorithms on scores produced and whether we'll need to test on the whole dataset.This knowledge can be helpful when training times are critical, especially on larger datasets.

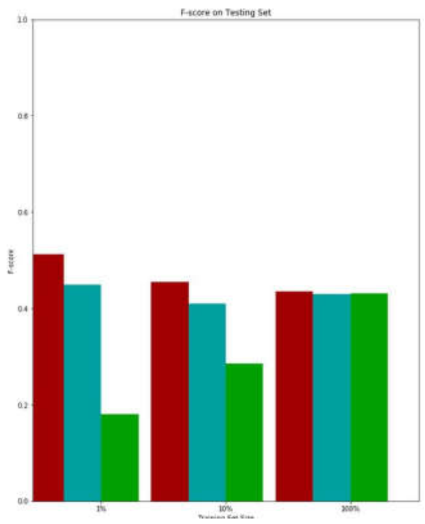
The first set of graphs trains using the **training set** to obtain the scores



The second set of graphs trains using the **testing set** to obtain the scores







A challenge on this section is the fact that knowing what algorithm to test-on requires both experience and intuition, depending on the type of data provided.

Refinement

Checking the values of time vs scores outputted, we decided to go with the decisiontreeclassifier. This algorithm had the least time while testing on both training set and test set, while producing nearly similar scores as the rest of the algorithms.

Looking at my use-case, the small difference in performance on time made a big difference when using cross-validation algorithm (ShuffleSplit) during training. This is the marginal advantage *decisiontreeclassifier* algorithm had over xgboost.

I used *GridsearchCV* algorithm to obtain the best possible algorithm *hyperparameters* which would be used in fitting the training data. This algorithm does an exhaustive search over specified parameter values. These are preset then fed to the algorithm.

The preset parameters were:

max_depth: list of integers that determine how many levels of descending branches a tree can grow to. List values: *[5,10,20]*

min_samples_leaf: list of integers which describe the minimum number of samples required per leaf node; where a leaf is a container of the subset data. List values: *[5,10,20,30,60,100]*

min_samples_split: list of integers describing the minimum number of samples required to split an internal node. List values: *range(10,500,20)* - the range between 10 to 500, with skip of 20 values in the list generated i.e. [10, 30, 50, …]

IV. Results

Model Evaluation and Validation

For validation, we used the *shufflesplit* algorithm, a type of cross-validation algorithm much like k-fold algorithm. K-fold algorithm involves creating a subset of data from the training set as the validation set. This is called a fold. This fold is created k-times sequentially over the full training set.

The advantage of using Shufflesplit is that these folds are selected at random on the dataset, and does not need to create folds over the full dataset i.e. takes a sample of dataset data instead of using the full dataset. This makes shufflesplit less time intensive compared to k-fold for non-trivial size dataset.

We tested the accuracy metrics on individual folds to check on the robustness of the algorithm employed, testing with on a list random states *[0,50,100,300,700,1000]* on the Shufflesplit algorithm. Ideally there should be very little variance (standard deviation) per fold, and marginal variance compared with the testing score. The testing score tends towards the maximum accuracy score of the calculated mean per fold; where *Max Acc Diff < Min Accuracy Diff*

```
Random state:0
******************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7680526315789473, 0.7631052631578947,
0.7666315789473684, 0.7648421052631579, 0.7665263157894737, 0.7668421052631579,
0.7636315789473684, 0.7641052631578947, 0.7641052631578947, 0.7680526315789473]
Accuracy: 0.7656 (+/- 0.00350)
Min Accuracy: 0.7621, Max Accuracy: 0.7691
Min Accuracy Diff 0.0060, Max Acc Diff: 0.0010
Test Accuracy tends towards Fold-Mean Max Accuracy

Random state:50
******************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7669473684210526, 0.7683684210526316,
0.7617368421052632, 0.7706842105263157, 0.7667894736842106, 0.766421052631579,
0.7695263157894737, 0.7660526315789473, 0.7637368421052632, 0.7629473684210526]
Accuracy: 0.7663 (+/- 0.00541)
Min Accuracy: 0.7609, Max Accuracy: 0.7717
Min Accuracy Diff 0.0071, Max Acc Diff: 0.0037
Test Accuracy tends towards Fold-Mean Max Accuracy

Random state:100
******************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7652105263157895, 0.7704736842105263,
0.7656315789473684, 0.7703684210526316, 0.7659473684210526, 0.7581578947368421,
0.7674210526315789, 0.7685263157894737, 0.7657894736842106, 0.7593157894736842]
Accuracy: 0.7657 (+/- 0.00784)
```

```
Min Accuracy: 0.7578, Max Accuracy: 0.7735
Min Accuracy Diff 0.0102, Max Acc Diff: 0.0055
Test Accuracy tends towards Fold-Mean Max Accuracy

Random state:300
*****************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7676315789473684, 0.7643157894736842,
0.759421052631579, 0.7653684210526316, 0.7653157894736842, 0.7602631578947369,
0.7642631578947369, 0.7649473684210526, 0.7693684210526316, 0.7608947368421053]
Accuracy: 0.7642 (+/- 0.00603)
Min Accuracy: 0.7581, Max Accuracy: 0.7702
Min Accuracy Diff 0.0099, Max Acc Diff: 0.0022
Test Accuracy tends towards Fold-Mean Max Accuracy

Random state:700
*****************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7648421052631579, 0.7630526315789473,
0.7599473684210526, 0.7689473684210526, 0.7608947368421053, 0.7650526315789473,
0.7663157894736842, 0.7663684210526316, 0.7688421052631579, 0.7626842105263157]
Accuracy: 0.7647 (+/- 0.00582)
Min Accuracy: 0.7589, Max Accuracy: 0.7705
Min Accuracy Diff 0.0092, Max Acc Diff: 0.0025
Test Accuracy tends towards Fold-Mean Max Accuracy

Random state:1000
*****************************************************************
Testing Score 0.7680526315789473
Accuracy for individual fold [0.7665263157894737, 0.7666315789473684,
0.7620526315789473, 0.7607368421052632, 0.7658421052631579, 0.7627368421052632,
0.763578947368421, 0.7661052631578947, 0.7644736842105263, 0.7661052631578947]
Accuracy: 0.7645 (+/- 0.00399)
Min Accuracy: 0.7605, Max Accuracy: 0.7685
Min Accuracy Diff 0.0076, Max Acc Diff: 0.0004
Test Accuracy tends towards Fold-Mean Max Accuracy
```

Final Model Produced

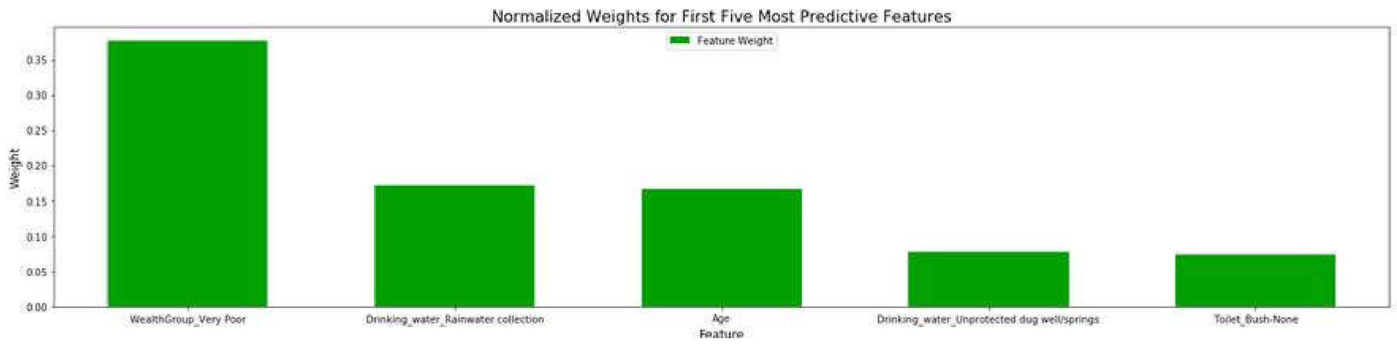
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=20,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=250,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Model Scores after Final Evaluation

Metric	Unoptimized Model	Optimized Model
Accuracy Score	0.7675	0.7675
F-score	0.4691	0.4691

V. Conclusion

Free-Form Visualization



This is a plot of important features by weight. This is particularly helpful when the featureset is very big and needs to be reduced, which has the advantage of being less computationally intensive and faster training times.

Analysis the data with the reduced featureset produces a score of

Accuracy on testing data: 0.7647

F-score on testing data: 0.4205

Compared to

Accuracy on testing data: 0.7675

F-score on testing data: 0.4691

As observed the produced scores of the reduced dataset are within range of the model trained on the full featureset. In our case since the total number of features are manageable (47 features), we used the full featurespace for training the model

Reflection

The model performs relatively well with the provided featureset with an accuracy of over 75%, compared with the naive prediction score obtained.

Setting the beta-score as 1.5, since we are emphasizing on recall, gives us an F-score of 0.4691. A low F-score can be interpreted as the model having the tendency of choosing from the majority class, since the dataset is imbalanced.

Since the F-score is at 0.4691, this score is not too low, therefore checking this score combined with the accuracy, we can conclude that the model properly classifies households into the respective groups on most use cases.

A major challenge experienced is improving the F-score. It was observed that the challenge is the data we are currently using, and increasing the feature-space might help in boosting the score.

Improvement

1) Use of XGBoost algorithm to improve score performance. Whilst for this project I chose the decisiontreeclassifier algorithm to train on, the XGBoost should perform better as this is an optimized distributed gradient boosting algorithm. In my case, using decisiontreeclassifier algorithm was much more time efficient on the hardware I was running

2) Increase the featureset. Having a smaller featureset has the effect of reducing model exploration during training.

3) Have a complete dataset. Some features and records had to be expunged because of missing data. This reduces the training set data, which reduces exploration of the model during training, which affects the final predictions produced

References

1. *Hunger Safety Net Programme, Cash Transfer Program*

http://hsnp.or.ke/

2. *Supervised Learning*

https://en.wikipedia.org/wiki/Supervised_learning

3. *Precision and recall*

https://en.wikipedia.org/wiki/Precision_and_recall

4. *Why we need social protection*

http://www.developmentpathways.co.uk/publications/why-we-need-social-protection/

5. *HSNP2 Impact Evaluation Final Report*

http://hsnp.or.ke/index.php/our-work/downloads

6. *Cash transfers: what does the evidence say?*

https://www.odi.org/sites/odi.org.uk/files/resource-documents/10749.pdf